Comparative Analysis of Dimension Reduction Techniques for Image Classification

Yuval Sheinin

Data Streaming Algorithms and Online Learning - Final Paper
Winter 2024

Abstract

This paper presents a comparative study of three prominent dimension reduction techniques: the Johnson-Lindenstrauss lemma [1], Principal Component Analysis (PCA) [2], and Autoencoders. These techniques are applied to the widely-used image dataset CIFAR-10 [3], aiming to enhance the performance of a Convolutional Neural Network (CNN) classifier, specifically ResNet18 [4], by striving to achieve comparable accuracy while requiring fewer dimensions. The quality of the reduced dimensions is assessed by reconstructing the original data and calculating the Mean Squared Error (MSE). Subsequently, the classification accuracy and computational efficiency of the CNN model are evaluated on the reduced-dimensional data. The findings reveal the strengths and weaknesses of each technique, offering insights into scenarios where one approach outperforms the others in terms of dimension reduction, classification performance, and computational efficiency.

1 Introduction

In the era of big data, high-dimensional datasets present challenges in storage, computation, and analysis. Dimension reduction techniques aim to mitigate these challenges by transforming data into a lower-dimensional space while preserving essential information. This paper examines three key techniques, namely the Johnson-Lindenstrauss lemma (JL) [1], Principal Component Analysis (PCA) [2], and Autoencoders.

The goal of this study is to conduct a comparative analysis of these techniques on the widely-used image dataset, CIFAR-10 [3], and to evaluate their impact on the performance of a Convolutional Neural Network (CNN) classifier, specifically ResNet18 [4]. The quality of the reduced dimensions is assessed by reconstructing the original data and calculating the Mean Squared Error (MSE). Additionally, the classification accuracy and computational efficiency of the CNN model are evaluated on the reduced-dimensional data.

1.1 Johnson-Lindenstrauss Lemma (JL)

The JL lemma guarantees that a set of points in a high-dimensional space can be embedded into a lower-dimensional space while preserving distances. Practical algorithms inspired by JL include the basic Johnson-Lindenstrauss Transform (JLT) [1], Sparse Random Projection (SparseRP) [5], Very Sparse Random Projection (VerySparseRP) [5], Fast Johnson-Lindenstrauss Transform (FJLT) [6], and Subsampled Randomized Hadamard Transform (SRHT).

1.2 Principal Component Analysis (PCA)

PCA is a statistical procedure transforming observations of correlated variables into linearly uncorrelated principal components. Widely used in image processing, neuroscience, and data visualization, PCA extracts the most significant features while reducing dimensionality [2].

1.3 Autoencoders

Autoencoders compress input data into a latent-space representation, reconstructing output to minimize input-output differences. Applied in anomaly detection, denoising, and dimensionality reduction, autoencoders efficiently encode input data.

2 Background

2.1 Johnson-Lindenstrauss (JL) Lemma

The Johnson-Lindenstrauss (JL) lemma is a cornerstone in the field of dimensionality reduction, providing a theoretical foundation for preserving pairwise distances when projecting high-dimensional data into a lower-dimensional space [1].

Theoretical Guarantees: The JL lemma asserts that for any set of n points in a high-dimensional Euclidean space, there exists a linear map that can project the data into a lower-dimensional space while approximately preserving the pairwise Euclidean distances between the points [1]. Formally, given a set of n points in a d-dimensional space, a small positive number ε and $0 < \delta < \frac{1}{2}$, there exists a mapping to a k-dimensional space (where k is proportional to $\log(n/\delta)/\varepsilon^2$) such that for all pairs of points x and y, the distance between x and y is preserved within a factor of $1 \pm \varepsilon$ [7].

2.1.1 Dimensionality Reduction Techniques

In this project, we implement and utilize several dimensionality reduction techniques based on the JL lemma, including the Johnson-Lindenstrauss Transform (JLT), Fast Johnson-Lindenstrauss Transform (FJLT) [8], Sparse Random Projection (SparseRP) [5], Very Sparse Random Projection (VerySparseRP) [9], and Subsampled Randomized Hadamard Transform (SRHT) [10]. Each of these techniques offers unique advantages in terms of computational efficiency and preservation of data structure, making them suitable for different types of high-dimensional data.

Johnson-Lindenstrauss Transform (JLT): The JLT is a dimensionality reduction technique that embeds high-dimensional data into a lower-dimensional space. Specifically, given n points in d-dimensional space and a small $\varepsilon > 0$, the JLT states that there exists a linear map $A : \mathbb{R}^d \to \mathbb{R}^k$ (where $k = O(\varepsilon^{-2} \log n)$) such that for all pairs of points x_i, x_j, the Euclidean distance is approximately preserved: $(1-\varepsilon)\|x_i - x_j\|^2 \leq \|Ax_i - Ax_j\|^2 \leq (1+\varepsilon)\|x_i - x_j\|^2$.

Fast Johnson-Lindenstrauss Transform (FJLT): The FJLT is an improvement over the JLT that reduces the computational complexity. It uses a randomized Fourier transform to ensure that sparse vectors are mapped to dense vectors, preserving distances. The FJLT is defined by the map $\Phi = k^{-1}PHD$, where P is a sparse random projection, H is the discrete Fourier transform, and D is a random reflection. The

FJLT can compute approximate distance-preserving embeddings efficiently, making it a practical choice for high-dimensional data. The time complexity of the FJLT is $O(d \log d + \varepsilon^{-3} \log^2 n)$, which is faster than the JLT's $O(nkd)$ [8].

Sparse Random Projection (SparseRP) and Very Sparse Random Projection (VerySparseRP) Sparse Random Projections (SparseRP) were introduced by Achlioptas for speeding up random projections. Projection matrix R is utilized with independent and identically distributed (i.i.d.) entries, achieving a threefold speedup by processing only one-third of the data [5].

Very Sparse Random Projections (VerySparseRP) further optimize computation by using $s \gg 3$, such as $s = \sqrt{D}$ or even $s = \frac{D}{\log D}$, where D is the data dimensionality. Data is sampled at a rate of $1/s$, suggesting that even fewer samples may be sufficient for accurate estimates, especially with approximately normal distributions [9].

Subsampled Randomized Hadamard Transform (SRHT) The subsampled randomized Hadamard transform (SRHT) constitutes a structured method for reducing the dimensionality of vectors, leveraging the properties of the Walsh–Hadamard matrix [10]. Its construction involves a wide matrix comprised of three components: a random diagonal matrix D with entries being independent random signs, an orthogonal Walsh–Hadamard matrix H scaled appropriately, and a random matrix R that selects a subset of coordinates from the original vector uniformly at random. The key aspect lies in its ability to preserve the geometric structure of entire subspaces, as demonstrated by its usage in algorithms for randomized linear algebra [10]. This preservation of geometry is fundamental for its application in various numerical tasks. The SRHT's efficacy is underscored by its ability to achieve optimal constants within its bounds, a feat facilitated by a novel proof methodology [10]. The construction of the SRHT matrix inherently relies on the orthogonal properties of the Walsh–Hadamard matrix and its specific scaling, making it a versatile tool for dimensionality reduction in mathematical applications.

These techniques are all based on the idea of random projections, which have been shown to preserve the structure of the data while significantly reducing its dimensionality [1]. They offer different trade-offs between computational efficiency and accuracy, making them suitable for different types of high-dimensional data.

2.2 PCA

[Put text here]

2.3 Autoencoders

[Put text here]

3 Related Work

Dimension reduction techniques have been extensively studied and applied in the field of image classification.

- **PCA-Based Pooling Method**: A recent study proposed a pooling method based on Principal Component Analysis (PCA) for image classification [11]. The PCA-based pooling method, termed PCA Pool, was designed to retain as much feature information as possible while reducing the feature dimension and parameters of a convolutional neural network (CNN). The method was tested with various CNN architectures on datasets including MNIST, CIFAR10/100, and SVHN. The results showed that PCA Pool could retain information in the pooling window better and improve the image classification accuracy compared to traditional pooling methods.

- **Convolutional Autoencoders**: Another study combined Convolutional Autoencoders with a Fully Connected Network for supervised dimensionality reduction and predictions [12]. The proposed methodology was found to be efficient in terms of parameter count and achieved competitive results against state-of-the-art deep learning methods. Furthermore, the resulting Latent Space, optimized for the classification task, could be utilized to improve traditional, interpretable classification algorithms.

These studies highlight the potential of dimension reduction techniques in improving the efficiency and accuracy of image classification tasks. However, there is still a need for a comprehensive comparison of these techniques, which is the focus of this paper.

4 Methodology

4.1 Datasets

The CIFAR-10 dataset was used in this study. CIFAR-10 consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

4.1.1 Streaming Case Dataset

In this study, we present our approach to dynamic dataset management, simulating online dataset streaming. We accomplish this by developing a suite of streaming dataset functions, which includes visualization utilities for data exploration, dynamic brightness adjustment to augment dataset diversity, and background processing for seamless dataset updates. Our methodology furnishes models with the capability to dynamically adapt to evolving data distributions and simulates a stream of endless new data to the model.

Specifically, our approach involves concurrent threads dedicated to updating both training and validation datasets at regular intervals. These updates involve the transformation of original images by multiplying them with a constant, thereby dynamically altering their brightness.

4.2 Dimension Reduction Implementation

4.2.1 Johnson-Lindenstrauss

The Dell JLT, a Python 3.x package, implements the Johnson-Lindenstrauss Transform (JLT) for dimensionality reduction [13]. It includes the original JLT and faster variants like the Fast Johnson-Lindenstrauss Transform (FJLT) and Randomized Hadamard Transform (RHT). These functions accept high-dimensional inputs, preserving pairwise distances with bounded error. The Dell JLT finds applications in linear map-

pings, random projections, hashing, and matrix approximations, offering efficient processing while maintaining data integrity [13].

4.2.2 Principal Component Analysis (PCA)

[Provide a detailed explanation of PCA, its underlying principles, and its implementation details.]

4.2.3 Autoencoders

[Provide a detailed explanation of Autoencoders, their architecture, and their implementation details.]

4.3 Classification Model

The ResNet18 architecture, a Convolutional Neural Network (CNN), was used as the classification model in this study.

Architecture: The ResNet18 model was initialized with default pre-trained weights. The model was then customized for the task at hand. The layers of the ResNet18 model were used for feature extraction. These layers were followed by an Adaptive Average Pooling layer to reduce the spatial dimensions to 1x1. The output of the pooling layer was then passed through a fully connected layer for classification. The number of output units in the fully connected layer was set to the number of classes in the CIFAR-10 dataset.

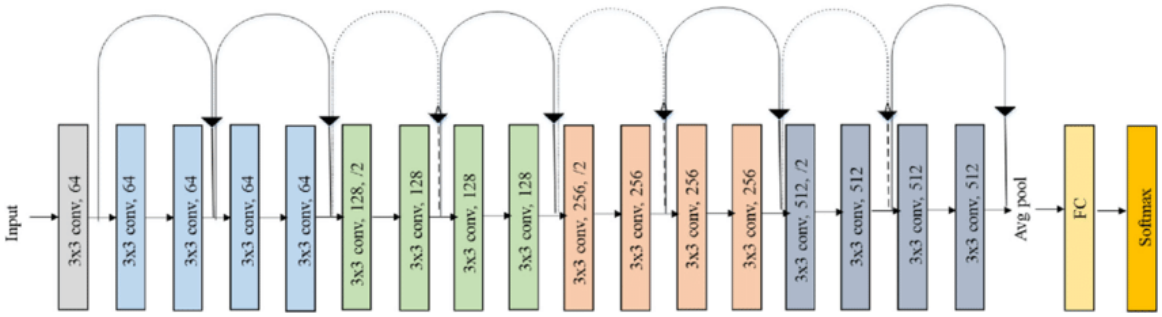


Figure 1: The architecture of the original ResNet-18 model.

Training Procedure: The model was trained using a custom function that iterated over training and validation phases across specified epochs. Within each epoch, parameter updates were based on computed losses from predictions and true labels, utilizing the Cross Entropy Loss for classification, and executed through the Adam optimizer. Gradients were computed and applied during training to update parameters, while validation ensured parameter integrity by omitting gradient computation. Throughout training, the function tracked running loss and correct predictions, enabling the calculation of average loss and accuracy metrics.

4.4 Evaluation Metrics

The quality of the reduced dimensions was assessed by reconstructing the original data and calculating the Mean Squared Error (MSE) between the original and reconstructed data. The classification performance was evaluated using accuracy as the metric. Additionally, the computational efficiency was measured by recording the running time of the classification model on the original and reduced-dimensional datasets.

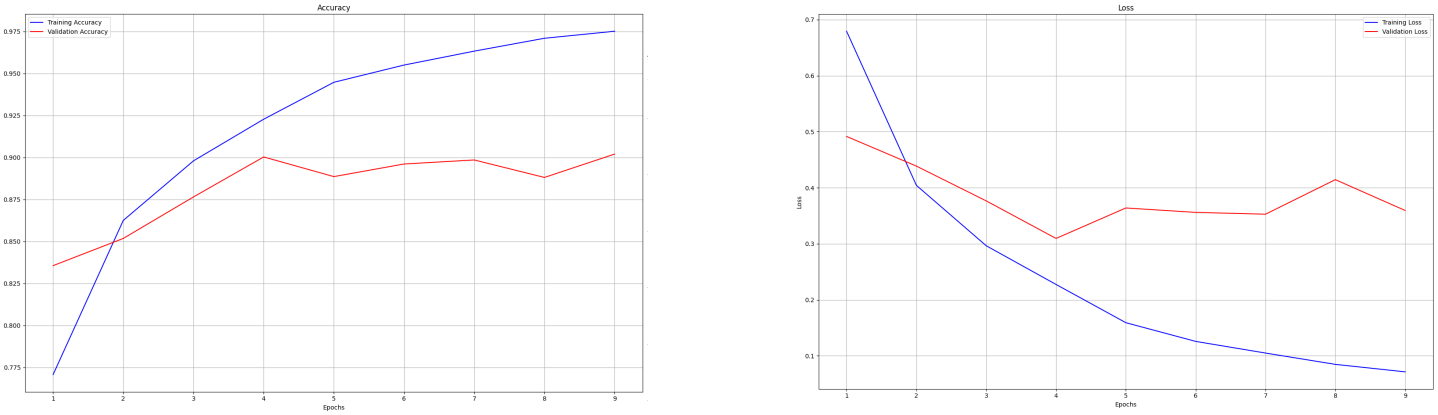
4.5 Results and discussion

4.5.1 JL Algorithms

We performed 6 tests:

1. **CIFAR-10 Dataset Fine-Tuning with ResNet18 - No Dimension Reduction:** Conducted fine-tuning on the original CIFAR-10 dataset without dimensionality reduction. The entire dataset was utilized to train the ResNet18 model, yielding the following results:

```
val Loss: 0.3648 | Acc: 0.8996
Training complete in 31m 45s
Best val Acc: 0.902000
```



(a) Accuracy as function of epochs

(b) Loss as function of epochs

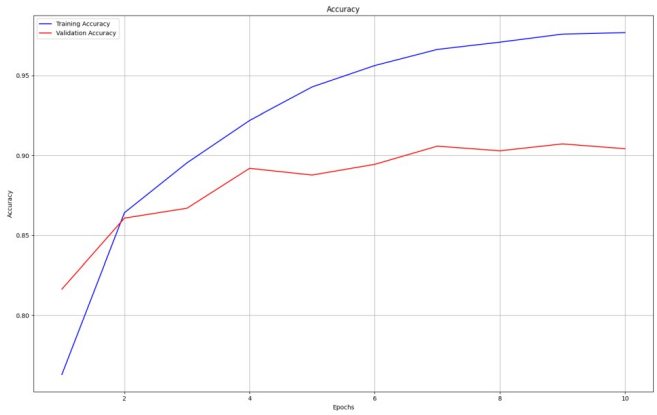
Figure 2: Fine tuning Resnet18 using constant CIFAR10

As expected from ResNet18, a state-of-the-art network, the validation outcome, surpassing the 90% mark, demonstrates exemplary performance. Analysis of the graphical representations (see Figure 1) elucidates the successful training process.

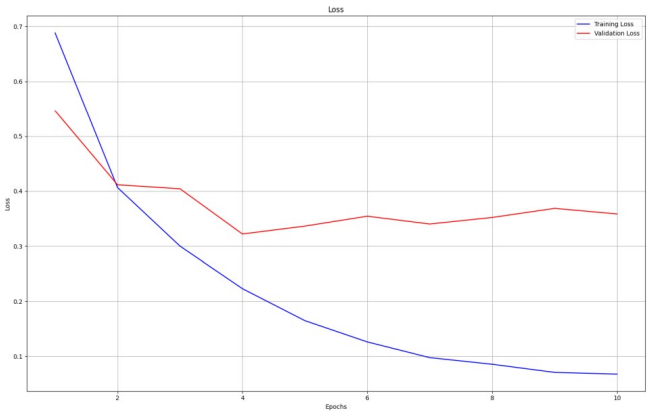
2. **Online CIFAR-10 Dataset Fine-Tuning with ResNet18 - No Dimension Reduction:** Conducted fine-tuning on the original classic CIFAR-10 dataset using ResNet18, leveraging the entire dataset. We utilized an online dataset, akin to the method explained in Section 4.1.1. However, for each batch, we randomly selected images and applied brightness changes to simulate an online stream. We abstained from using threads to conserve computational resources. These adjustments resulted in the following outcomes:

```
val Loss: 0.3585 | Acc: 0.9043
Training complete in 32m 7s
Best val Acc: 0.907300
```

We observe that despite implementing a brightness change on 15% of the images in each batch, where the brightness change factor is randomly distributed between 0.8 and 1.2, ResNet18 did not significantly react to this variation, and the validation accuracy did not deteriorate.



(a) Accuracy as function of epochs



(b) Loss as function of epochs

Figure 3: Fine tuning Resnet18 using constant CIFAR10

3. **CIFAR-10 Dataset Fine-Tuning with ResNet18 - Dimension Reduction Using Dell/jlt:**

To begin, we initially applied Johnson-Lindenstrauss (JL) transforms directly to input images before processing them through ResNet18. However, this approach yielded unsatisfactory results due to the loss of crucial information, such as special features and pixel order, rendering it incompatible with convolutional neural networks like ResNet.

In response, we developed a dedicated layer specifically tailored for JL transforms, seamlessly integrating it into the ResNet pipeline. This layer strategically operates within the latent feature space, positioned between the output of the feature extraction module and the input to the ResNet18 fully connected layer, resulting in better results.

Moving forward, we conducted a comprehensive evaluation of each Dell Johnson-Lindenstrauss method, exploring various components and generating accuracy graphs to assess their performance.

The results are presented in Appendix 1, where it becomes evident that the accuracy levels across different methods were remarkably similar, even strikingly so. It is observed that as the number of components increases, the accuracy also improves, which aligns with expectations since less information is lost. Interestingly, it is noted that reducing the number of components from 512 to 200 does not significantly impair accuracy. However, such reduction in components results in a 10% decrease in running time, as we will see later.

Additionally, we performed a computational analysis focusing on runtime and memory consumption across different JLT methods, with a fixed number of components set at 256.

Remarkably, it is evident that SRHT surpasses FJLT in terms of execution time.

Furthermore, we extended our investigation to the Fast Johnson-Lindenstrauss Transform (FJLT), conducting experiments to measure both time and accuracy across various component configurations. Each run was performed for 5 batch and averaged.

Method	Peak Memory (MB)	Execution Time (sec)
FJLT	6.54	33.26
JLT	5.14	156.48
SparseRP	2.10	110.58
VerySparseRP	0.80	102.46
SRHT	0.66	23.24

Table 1: Comparison of JL methods for fixed number of components (256)

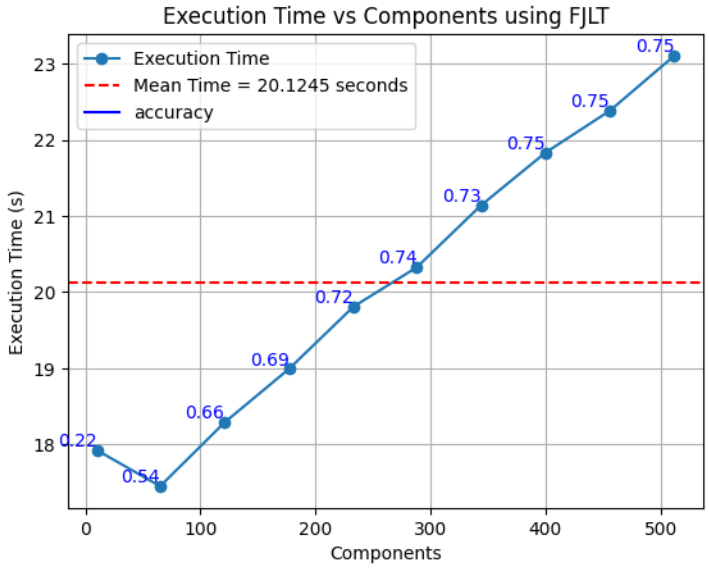


Figure 4: Execution Time vs Components using FJLT

When we attempted to measure the execution time without any dimension reduction layer at all, over 5 epochs and averaging the results, we obtained the following outcome:

```
Best val Acc: 0.654
Training with FJLT took 15.3649 seconds.
```

Upon summarizing both the graphical representation and the numerical findings, it becomes apparent that increasing the number of components leads to higher accuracy, as depicted in the graph. However, it is noteworthy that maximum accuracy does not necessitate a maximum number of components, such as 512. Achieving almost maximum accuracy is attainable with approximately half the dimensions used by the original network.

When considering the numerical results obtained without any dimension reduction, an accuracy of approximately 65% was achieved with a runtime of 15.36 seconds, indicating that this runtime represents the fastest scenario. From a runtime perspective, it is advantageous to avoid dimension reduction as it introduces additional processing time.

However, it is important to note that employing dimension reduction, particularly after five epochs

and averaging over the same, results in increased accuracy, surpassing that of the original network while utilizing fewer dimensions. Notably, starting from 100 components out of 512, the network demonstrates superior accuracy compared to the original.

The provided graph exhibits a monotonically increasing trend, illustrating that higher component counts correlate with longer runtimes and improved accuracy. Additionally, it is noted that the accuracy saturates before reaching the maximum number of components. Beyond a certain threshold, further increasing the number of components does not yield a proportional increase in accuracy, suggesting that approximately half of the components can be discarded while maintaining accuracy levels comparable to those achieved with the complete set.

4. **Sparse Random Projection on CIFAR10 Dataset with SGD Classifier (log_loss = Logistic Regression):** This experiment closely resembles our work with the MNIST dataset in class, except for the utilization of logistic regression.

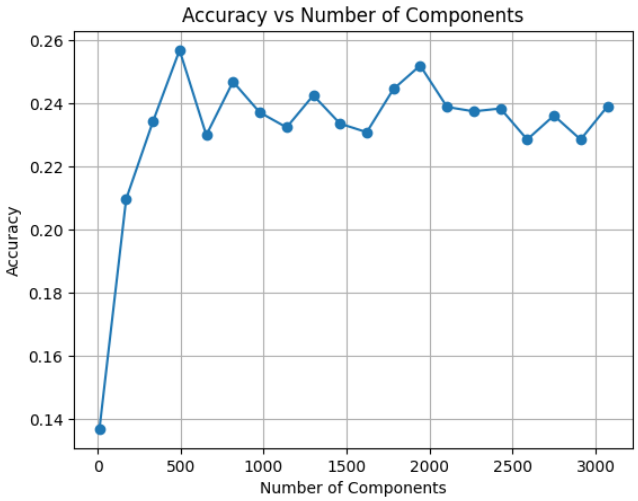
   We conducted an analysis to examine the impact of dimensionality reduction on classification accuracy by employing Sparse Random Projection (SRP) combined with an SGD classifier utilizing log loss. The process begins by initializing a list to store accuracy values and generating a range of component values for SRP using np.linspace().

   Subsequently, the code iterates through each component value, applying SRP to transform both the training and testing data accordingly. Within each iteration, the SGDClassifier is initialized with log loss, and training is performed over a fixed number of epochs using partial_fit(). Following each epoch, the accuracy on the test set is computed and the results are accumulated.

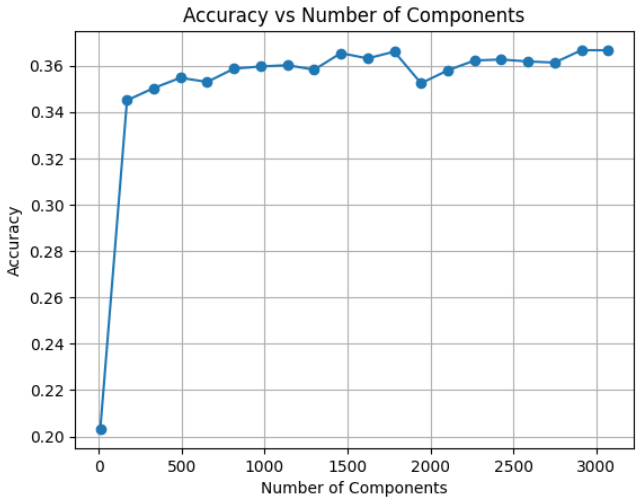
   The average accuracy over the epochs is then calculated and appended to the accuracies list. Finally, the code plots the relationship between the number of components and the average accuracy using matplotlib. This visualization aids in understanding how varying dimensionality impacts classification performance and provides insights into the trade-offs between dimensionality reduction and classification accuracy.

   In both graphs, it is evident that the accuracy is notably low. Even when utilizing the optimal case optimizer, we only observe slightly improved results. Our rationale for these poor outcomes is rooted in the complexity of the CIFAR10 dataset. Unlike MNIST, CIFAR10 poses a significantly greater challenge due to its intricate nature. Linear models struggle to adequately capture the complexities inherent in such datasets. Consequently, even with an increase in the number of components, the accuracy does not demonstrate any significant enhancement. Hence, employing dimensionality reduction techniques like Sparse Random Projection along with Logistic regression fitting proves to be ineffective and is not recommended for handling datasets of such complexity.

Dimensionality Reduction Experiments 5 and 6: Employing JL Transformations and Sparse Random Projection Techniques on CIFAR-10 Dataset via Convolutional Neural Network Fitting: In both Test 5 and Test 6, we aimed to enhance the performance of a 1D deep convolutional neural network (CNN) for classifying images from the CIFAR-10 dataset using different dimensionality reduction methods. However, the results remained consistently poor. In Test 5, we constructed a CNN1D model and applied dimensionality reduction using Sparse Random Projection



(a) Classification Accuracy for Adaptive Case



(b) Classification Accuracy for Optimal Case

Figure 5: Classification Accuracy Comparison

(sparseRP) methods from sklearn, alongside training with cross-entropy loss. Despite these efforts, the model struggled to extract meaningful features, resulting in low training and validation accuracies of around 10% and 8%, respectively.

In Test 6, following the disappointing outcomes of Test 5, we made adjustments to the CNN1D architecture and employed dimensionality reduction using JL Transformations by Dell. Specifically, we focused on modifying the input size of the first fully connected layer. Despite these alterations, there was no improvement in performance. Accuracies stagnated at similarly low levels, indicating the need for further exploration and refinement of the model's architecture and training configurations.

Overall, both tests underscored the limitations of the selected model architectures and training strategies in effectively addressing the classification task. The CNN alone was not strong enough to find patterns in the dimension-reduced data, likely due to the destructive nature of dimensionality reduction techniques on specialized data.

4.5.2 Autoencoders

[Present and discuss the results of Autoencoders techniques, including dimension reduction quality, classification performance, and computational efficiency.]

4.5.3 PCA

[Present and discuss the results of PCA techniques, including dimension reduction quality, classification performance, and computational efficiency.]

5 Conclusion

[Summarize the key findings of the study and provide concluding remarks.]

In summary, regarding JL techniques, our empirical investigations indicate that employing Johnson-Lindenstrauss (JL) transformations within the visual domain for classification tasks did not yield optimal efficiency. Instead, employing JL transformations within the latent space proved to be more effective. This preference arises from the inherent property of JL transformations to preserve pairwise distances in reduced dimensions while concurrently obliterating specialized information, which holds crucial significance for convolutional neural networks (CNNs).

A notable discovery emerged from Test 3, where Figure 3 visually depicted that despite an approximate 10% increase in execution time, higher accuracy could be achieved through the utilization of fewer dimensions within the ResNet architecture. This finding underscores the nuanced relationship between computational cost, dimensionality reduction, and classification accuracy, providing valuable insights for architectural optimizations in CNNs.

6 Future Work

[Discuss potential avenues for future work, such as exploring hybrid dimension reduction techniques, investigating more datasets and classification models, or extending the study to other application domains.]

7 References

[List all the relevant references cited in the paper.]

References

[1] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[2] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[3] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 287–296.

[6] N. Ailon and B. Chazelle, "The fast johnson–lindenstrauss transform and approximate nearest neighbors," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 302–322, 2009.

[7] M. Skorski, "Johnson-lindenstrauss transforms with best confidence," vol. 134, pp. 3989–4007, 15–19 Aug 2021.

[8] O. N. Fandina, M. M. Høgsgaard, and K. G. Larsen, "The fast johnson-lindenstrauss transform is even faster," *arXiv preprint arXiv:2204.01800*, 2022.

[9] C. Chen, C.-M. Vong, C.-M. Wong, W. Wang, and P.-K. Wong, "Efficient extreme learning machine via very sparse random projection," *Soft Computing*, vol. 22, pp. 3563–3574, 2018.

[10] J. A. Tropp, "Improved analysis of the subsampled randomized hadamard transform," *Advances in Adaptive Data Analysis*, vol. 3, no. 1-2, pp. 115–126, 2011.

[11] B. Zhao, X. Dong, Y. Guo, X. Jia, and Y. Huang, "Pca dimensionality reduction method for image classification," *Neural Processing Letters*, vol. 54, pp. 347–368, 2021.

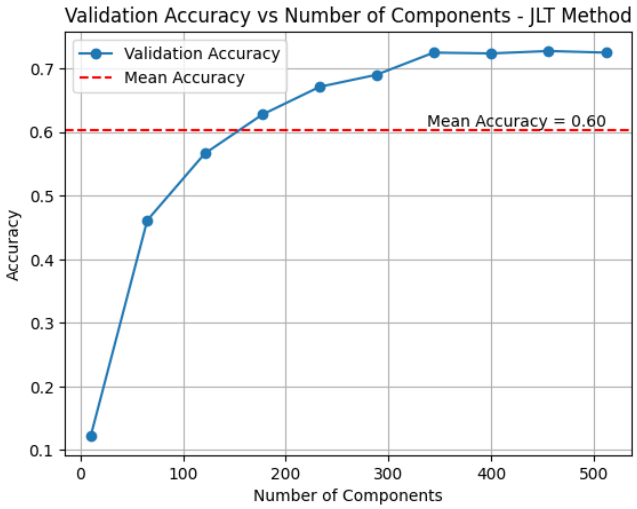
[12] Y. Liu, C. Ponce, S. L. Brunton, and J. N. Kutz, "Multiresolution convolutional autoencoders," *arXiv preprint arXiv:2004.04946*, 2020.

[13] B. P. Fauber, "Johnson-lindenstrauss transforms," https://github.com/dell/jlt, 2023.

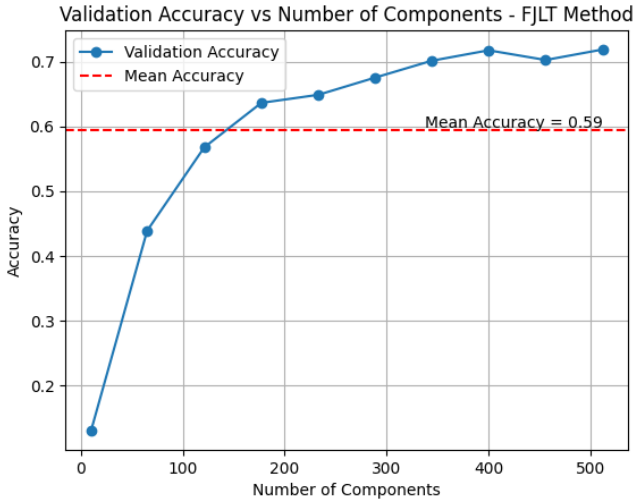
8 Appendices

[Include any additional materials, such as code snippets, visualizations, or supplementary results, as appendices.]

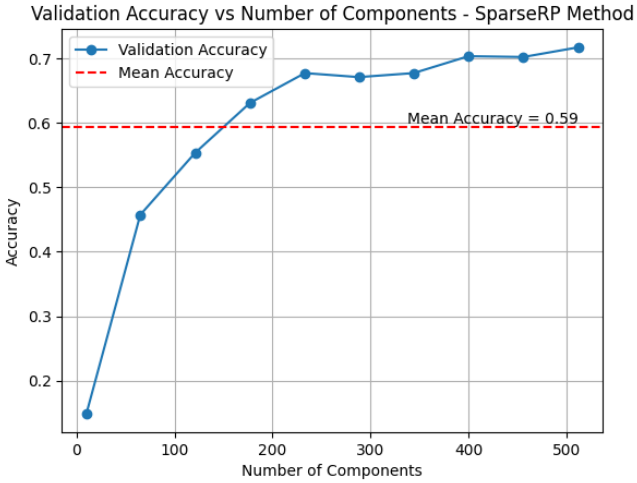
Appendix 1: Comparison of JL Methods



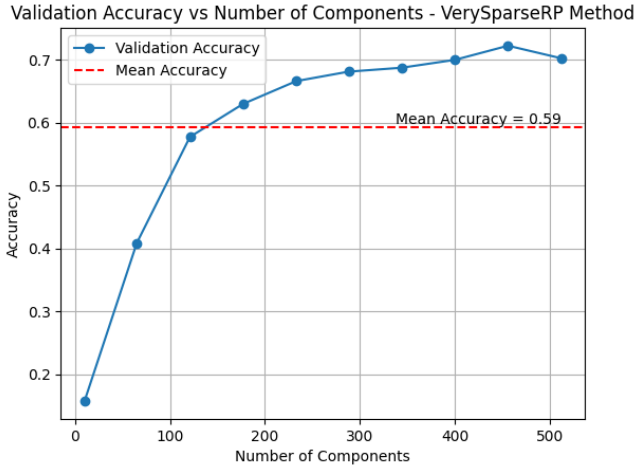
(a) JL with JLT



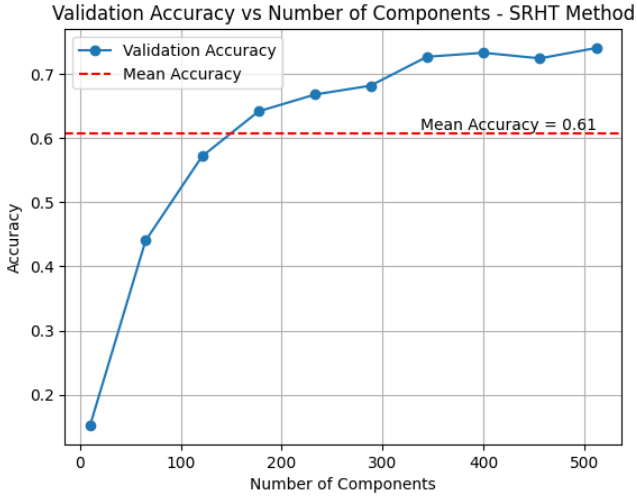
(b) JL with FJLT



(c) JL with SparseRP



(d) JL with VerySparseRP



(e) JL with SRHT

Figure 6: Comparison of JL Methods